The Current State of Face Tracking for Video Surveillance

[1]Chris Whiten, [2]Wassim Bouachir, [2]Guillaume-Alexandre Bilodeau, [1]Robert Laganière
[1]*University of Ottawa, Ottawa, Canada*
[2]*École Polytechnique de Montréal, Montréal, Canada*
[1]{*cwhit025, laganier*}*@uottawa.ca,* [2]{*wassim.bouachir, guillaume-alexandre.bilodeau*}*@polymtl.ca*

Abstract

A fundamental task in video surveillance is that of visual tracking, specifically in environments where surveillance is a priority, such as airports. Many approaches to tracking have been proposed, with different drawbacks and advantages that affect their suitability for face and human tracking for surveillance. In this report, we present the most relevant and recent methods for tracking and evaluate their suitability for the aforementioned task.

1. Introduction

Person tracking is a fundamental problem in computer vision and visual surveillance. Automated person tracking permits more complex tasks such as human activity/event recognition (for example, recognition of suspicious activities), scene analysis (for example, computing the number of visitors to a specific area), and calculation of face logs (for example, storing multiple views of the same person's face, for future analysis tasks such as face recognition).

Tracking can be defined as, given a target to track, to localize that target in each consecutive frame based on a combination of its previous position and its stored appearance. This yields the computational benefit of not having to detect the target by an entire image search (by leveraging its previously tracked location), and allows us to infer a target's location when detection fails or it becomes occluded. By tracking a target throughout a sequence, we can extract its trajectory through time. Often times, it is a requirement that this be computed in real-time due to the continuous and ongoing nature of surveillance footage, which means that a tracker may only use information from the present and past video frames, and is not able to look into future frames for target appearance. Another common requirement is that when the target leaves the camera's field of view and returns, the same identity must be associated with the target before leaving and after returning to the field of view. As an example, if a tracker is employed to count the number of visitors to a room within a day, the same visitor should only be counted once, regardless of whether the camera is able to see that visitor at all times.

Due to its occurrence in several more complex computer vision tasks, tracking has been thoroughly studied both in the general domain and for specific applications (surveillance, human-computer interaction, navigation). The goal of this survey is to evaluate and compare recent approaches that are applicable to human and face tracking in a surveillance environment, for different pre-defined scenarios. These scenarios are thoroughly described in Section 4. Briefly, we are interested in trackers applicable to tracking an isolated face in a constrained environment (person in lane), trackers applicable to tracking a person entering and exiting a controlled doorway (controlled chokepoint) and an uncontrolled doorway (uncontrolled chokepoint), trackers applicable to tracking multiple people in both general sparse and dense environments,

Paper	Histograms	Local Features	Sparse-representation	Learned Appearance
Sevilla-Lara et al. [45]	yes	no	no	no
Oron et al. [39]	es	no	no	no
Babenko et al. [7]	no	yes	no	yes
Adam et al. [2]	yes	no	no	no
Erdem et al. [17]	yes	no	no	no
Woo Park et al. [49]	no	yes	no	yes
Shu et al. [46]	no	yes	no	yes
Avidan [6]	yes	yes	no	yes
Mei et al. [36]	no	no	yes	no
Mei et al. [37]	no	no	yes	no
Zhang et al. [52]	no	no	yes	no
Zhong et al. [54]	yes	no	yes	no
Andriyenko et al. [5]	yes	no	no	no
Jia et al. [23]	no	no	yes	no
Li et al. [31]	no	yes	no	no
Kalal et al. [25, 26]	no	yes	no	yes
Yang et al. [50]	no	yes	no	yes
Hare et al. [21]	no	yes	no	yes
Comaniciu et al. [14]	yes	no	no	no
Kang et al. [27]	yes	no	no	no
Wagner et al. [48]	no	yes	no	no
Zhou et al. [55]	yes	yes	no	no
Liu et al. [32]	yes	no	no	no

Table 1. Comparison of appearance models used in recent approaches

and trackers applicable to tracking multiple people in an unrestricted outdoors environment. For a thorough survey of less recently-proposed methods, readers are referred to a tracking survey by Yilmaz et al. [51].

We separately identify different appearance models in Section 2 and different motion models in Section 3. As these approaches are described, we introduce and compare similar tracking methods that use these motion and appearance models. Finally, we evaluate their readiness based on the TRL (technology readiness level) for different surveillance-based scenarios in Section 4 and conclude in Section 5.

2 Appearance Models

Arguably the most important aspect to any tracker is its appearance model. The appearance model largely dictates the successes and shortcomings of trackers, defining the feature-space used to represent the target. Sometimes we trade off keeping information about one domain of the target in exchange for robustness to certain criteria, or computational efficiency. For example, a commonly-used construct in appearance models are histograms, which lose spatial information in exchange for point-to-point transformation invariance and resistance to motion blur. In this section, we describe some of the most common appearance models in the recent literature, and a tabular comparison of some methods is presented in Table 1.

2.1 Template Matching

In template matching, a pre-computed region of interest containing the target (called a template) is stored and used as a model to compare potential candidate targets against. Each candidate is compared against the stored template by some similarity measure, such as a pixel-by-pixel comparison with the sum-of-squared-difference similarity measure [20], or cross-correlation [53]. These two methods use colour or intensity features and are popular approaches due to their simplicity and computational efficiency.

While tracking with a single template performs well when tracking static objects with little-to-no appearance change, this strategy is often insufficient for video surveillance tasks, where the target undergoes significant appearance changes due to phenomena such as human articulation. To account for this, either multiple templates must be stored or the template must be sequentially updated to reflect the current target appearance. When the target is unknown a priori, the former is not possible and the target must be updated online. The problem of deciding when to update a template throughout a tracking sequence remains an open problem. If a target's appearance changes throughout a scene, the template model should reflect those changes or it risks becoming out-of-date and the tracker will drift [34]. For example, if we are tracking a human's face and the human rotates at a 90 degree angle, the template will be out-of-date and match poorly against the actual target. A gradual update as the human rotates their head would ensure good matches and accurate tracking. However, updating the template at the wrong frame risks introducing artifacts into the template. For example, if the target is moving quickly at the time of update, motion blur will be included in the template and the template will be less accurate. Updating at the wrong frame also risks introducing occluders or background pixels to the template. Many different strategies have been proposed to tackle this problem. Matthews et al. [34] present a template update scheme that corrects for tracking drift by storing two templates: the template from the first frame and the most recent tracking result. Drift is corrected via gradient descent minimization of a cost score for aligning the current tracking result template with the first frame template. Many sparse representation trackers [36, 23, 52], described in Section 2.4 give a dynamic importance weight to each template, and update the template set by discarding the least important template and adding the current tracking result when the tracking reconstruction error is sufficiently high, up to some threshold.

When tracking with a single template, illumination changes in the scene often cause there to be large differences in the pixel values between the true target and the stored template. A standard approach for solving this issue is to use normalized cross-correlation [53], which normalizes the image by subtracting the mean value and dividing by the standard deviation. The resultant value is a similarity measure between a template and candidate target, computed by

$$similarity(c,t) = \sum_{x,y} \frac{(c(x,y) - \bar{c}) * (t(x,y) - \bar{t})}{\sigma_c \sigma_t} \tag{1}$$

The idea of template matching has been expanded in recent years. Rather than doing a pixel-by-pixel comparison, in Section 2.2.1 we explore methods that break a template into rectangular subregions and use histogram comparison over these subregions as a similarity measure. In Section 2.4, we explore methods that keep a large dictionary of templates and represent the current tracked target as a sparse linear representation of the template dictionary.

2.1.1 Contour Tracking

While a great deal of research has gone into tracking by a template represented by a geometric shape, such as an elliptical or rectangular region of interest, there also exist techniques that directly track the contour of a template. This has the benefit of excluding all background pixels from the template, which increases the precision of the pixel distribution and increases the discriminative power of the target representation.

Contour tracking aims to estimate the target's location, scale, and shape in each frame using either the contour from the previous frame or an external contour detector. The target's appearance is generally modelled by either a descriptor representing the shape's silhouette [9] or a colour/intensity histogram of the interior region of the shape boundary [27].

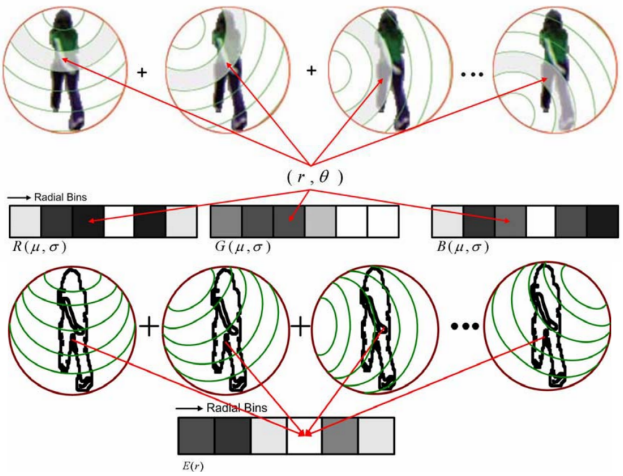

Figure 1. A shape contour is represented by a set of concentric circles, each of which provides a probability distribution over the contour edge pixels [27]

When matching directly by shape, there are two required steps. First, the shape of the target must be segmented from the overall image, either by background subtraction or an image segmentation algorithm. Then, a shape matching algorithm is employed to return the similarity between the template contour and the segmented shape. Shape Contexts [9] are among the most popular shape descriptors for matching. Points are randomly selected along the contour and a *shape context* is computed for each of those points. A shape context encodes the general spatial distribution of a given point relative to the remaining points on a contour with a log-polar space histogram. Then, a mapping is found between the shape context points from the template to the candidate target, which returns a similarity cost which we aim to minimize.

Kang et al. [27] propose a contour-based appearance model based on circular regions around the contour. The smallest circle that fully contains the contour is taken as a reference circle, and small concentric circles of various radii are extracted from within the reference circle to be used as bins. This is illustrated in Figure 1. Within each bin, the contour is modelled by a Gaussian colour model over the RGB distribution of contour edge pixels within that bin. The final shape descriptor is obtained by counting the number of contour-edge pixels within each bin, yielding a probability distribution. Then, any probability distribution comparison technique (such as the Kullback-Leibler distance), can be used as a similarity function between a candidate target and a template. This approach is invariant to changes in scale, rotation and translation.

2.2 Histograms

Histograms are a popular choice due to their simplicity and power. There exist methods to efficiently compute arbitrarily large histograms over arbitrary regions in an image in constant time [40] and several approaches are available to compare histograms. One of the major drawbacks to histograms is that all spatial information within the searched region is lost, since histograms are essentially a discretized sum of values over the image region. Some trackers [2, 17] combat this by computing histograms of neighbouring regions and enforcing that a pair of matching histograms must also have matching neighbourhood histograms. Other drawbacks of histograms include sensitivity to illumination changes, large appearance changes, and occlusions. Despite these drawbacks, histograms remain among the most popular appearance models in use.

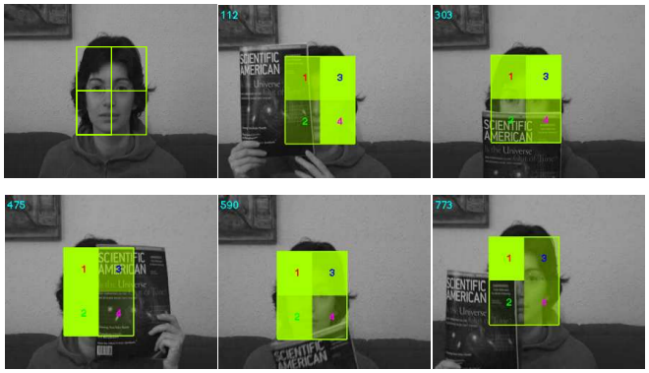

Figure 2. A reliability value (proportional to the shading) is assigned to each fragment. The reliabilities are dynamically updated at each frame and contribute proportionally to the final tracking result. Low-reliability (occluded) fragments have minimal impact on the tracking result [17]

2.2.1 Part-based histogram techniques

An alternative to tracking by a holistic template is to track by template fragments and then enforce spatial constraints on those fragments. Adam et al. [2] introduce Fragtrack, a tracking algorithm that permits real-time target tracking by breaking a target up into several small patches and comparing histograms on individual patches as a similarity measure. Histograms are efficiently computed using a data structure called the *integral histogram* [40], which is designed to compute histograms over arbitrary rectangular regions with a constant number of arithmetic operations. Fragtrack operates on a frame-by-frame basis by searching the entire image within a search radius r of the previously tracked location. For each candidate location c, a region of interest is extracted and broken up into several patches to compare against the template image t. The similarity between c and t is computed by comparing the histogram similarities between each extracted patch and sorting them. Then, the top 25% most similar patches are thrown away and the next most similar patch is used as the similarity score between c and t. Such a similarity measure grants the algorithm robustness against occlusions of up to 75% of the target template, making for a powerful tracker. However, the tracker is sensitive to full occlusions, low frame rates, and no appearance model update scheme is proposed.

Fragtrack was expanded upon by Erdem et al. in [17]. A particle filter framework is used for improved search efficiency, rather than exhaustive region searching. The particle filter framework also allows for more natural inclusion of scale information into the problem formulation. Tracking accuracy is improved via adaptive cue integration, which assigns a reliability to each of the "cues" (fragments or patches) and uses these to weight their contribution to the joint tracking result. Cues with low reliability contribute little to the final tracking result, while cues with high reliability greatly influence the final tracking result. A cue's quality is estimated with a sigmoid function of the Euclidean distance between the result suggested by an individual cue, and the final tracking result agreed upon by all cues. The reliabilities are updated at the "update" phase of the particle filter framework, via a technique called Democratic Integration [41]. Democratic Integration suppresses fragments that are not in agreement with the joint tracking result, while giving higher future influence to fragments that are in line with the joint result. A visualization of the cue reliabilities can be seen in Figure 2, where more transparent fragments correspond to low reliabilities and opaque fragments correspond to high reliabilities.

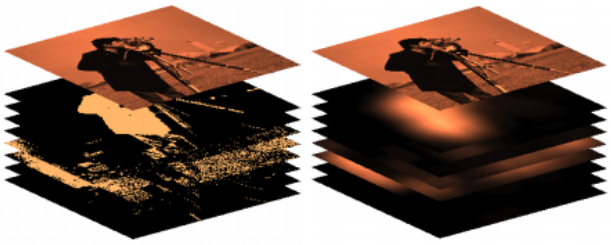

Figure 3. Distribution Fields explode an image into multiple layers, one for each histogram bin. These layers are shown on the left, and shown spatially smoothed on the right.

2.2.2 Holistic histogram techniques

Oron et al. [39] propose a tracking technique based on locally orderless matching (LOM), where an online learning approach allows the tracker to shift between rigid template matching and nonrigid histogram matching with Earth Mover's Distance [43] as a histogram matching metric. LOM allows the calculation of the likelihood of an image patch P being a noisy replica of a previous patch Q, $Pr(P|Q, \Theta)$, where Θ are noise model parameters. These noise model parameters control (in an online fashion) the cost of spatially moving pixels and the cost of changing a pixel's appearance. As these costs are adjusted, the tracker can tend to increase the likelihood of a rigid template or the likelihood of a deformable template. The paper shows that computing the MAP estimate of a mapping between the a patch P and its assumed replica patch Q is proportional to computing the optimal "signature Earth Mover's Distance", for a signature where the source is the pixels of the source patch, the destination is the pixels of the destination patch, and the distance function is defined as $d(p, q) = -\log(Pr(p|q, \Theta))$, for pixels $p \in P$ and $q \in Q$. Finally, tracking is formulated in a particle filter framework, where the particle likelihoods are inferred using the LOM described above. The noise model parameters Θ are learned in an online fashion, propagated through frames. The EMD flow between the final candidate particle's signature and the target signature is used to estimate the maximum likelihood of the noise model parameters, which are regulated using a prior and a moving average process. Generally, we compute the ML parameters of Θ_n, regularize them, then mix them in a moving average with the previous noise model parameters, Θ_{n-1}. This approach is very robust for handling a variety of targets, but suffers from a low tolerance for occlusions, no online model updating, and no approach to track multiple targets.

Distribution Fields [45] have been introduced as a template-based approach to object tracking. Tracking is performed by comparing the ℓ_1 difference between a model and candidate region, and following the direction of steepest descent until a local minimum is reached. In standard practice, this method of convex optimization of tracking works poorly due to low landscape smoothness (smoothness of the optimization space) and poor specificity (when the global optimum of a matching function does not represent the true position of the target). Distribution Fields alleviate these problems by exploding a frame into multiple layers, one layer for each binned feature, as seen in Figure 3. Each layer is spatially smoothed with a Gaussian, resulting in a probability distribution at each pixel corresponding to the probability that that location contains the layer's feature. This gives a smooth optimization space in the spatial dimension, but not the feature dimension. The image is then smoothed in the feature dimension, permitting a compact representation of all "neighbour images" (that is, all images corresponding to subtle, subpixel differences in motion, shadows, and changes in illumination). This increases the width of the capture range around the position of the tracked target, and gives rise to a very accurate tracker. However, there is no model for occlusion or variance in object pose, limiting the possible domains for this accurate tracker to be used.

2.3 Local Feature Tracking

Another popular method of tracking is to use specific local features, such as salient keypoints, and to track their displacements from frame to frame. We begin with a brief overview of some common feature choices for tracking, and then present some proposed approaches.

2.3.1 Keypoints

SIFT [33] is arguably the most famous descriptor to date. A grid of HOG (histogram of oriented gradients) features determines a 128 dimensional descriptor which is highly discriminant and robust to rotation, scale and location changes. Although SIFT has become the benchmark for keypoint matching algorithms, computing and comparing SIFT points is notoriously slow [12], leading researchers to investigate more compact descriptors that can be compared efficiently [8, 12, 30, 4].

SURF [8] is a popular alternative to SIFT with faster matching speeds. It uses the integral image to compute a 64 or 128 dimensional descriptor based on Haar-wavelets and local gradient histograms. Despite its increase in speed over SIFT, SURF descriptors are still represented by, at minimum, 64 floating point values. For real time applications, such as tracking, more compact representations are needed to minimize space and comparison runtime. These requirements have encouraged the advent of binary descriptors for keypoint representation.

Binary descriptors have become very popular due to their efficiency to compute, efficiency to compare, and compactness. A simple binary string is used to describe the keypoint, which is much more compact than a 128-dimensional vector of floating point values. Furthermore, two binary strings can be quickly compared by a simple Hamming distance calculation (XOR, followed by counting the number of 1 bits).

Binary Robust Independent Elementary Features (BRIEF) [12] descriptors describe an image patch with a binary string for the aforementioned reasons. The binary string is built by simple smoothed pixel comparisons. If we have two pixels to compare whose intensity values are denoted $p(x)$ and $p(y)$, the descriptor bit for these two pixels is set to 1 if $p(x) < p(y)$, which is trivial to compute and compact as a representation. Two BRIEF points are efficiently compared by computing the Hamming distance between their bit strings. Although this descriptor is powerful due to its efficiency and compactness, it suffers from a low tolerance to in-plane rotation and scale changes.

Binary Robust Invariant Scalable Keypoints (BRISK) [30] build on this idea and provide invariance to rotation and scale changes. Keypoints are detected by a variation of the FAST detector [42] that identifies keypoints based on their saliency along multiple scales through a scale-space pyramid. This scale-space search provides the descriptor with scale invariance. The descriptor is built by sampling a number of points around the detected keypoint in a specific circular sampling pattern, with higher sampling density closer to the keypoint. The sampled points are then separated into two sets, long-distance pairings and short-distance pairings. The long-distance pairings are used to estimate the direction of the keypoint, which is then used to rotate the sampling pattern, yielding rotation invariance. Then, the short-distance pairings are used to build the descriptor by simple intensity comparisons, similar to BRIEF [12].

Fast Retina Keypoints (FREAK) [4] are efficient binary keypoint descriptors based on a sampling pattern derived from the human retina. Matching is efficiently performed with these keypoints in a similarly biologically inspired manner. Keypoints are defined by comparing local pixels, where the sampling is dense around the center pixel and more sparse (corresponding to a human's peripheral vision) further away. Searching for a corresponding keypoint is first performed with only the "peripheral" bits from the keypoint, eliminating approximately 90% of the search space, which is then examined with the more densely sampled bits to find a match.

2.3.2 Tracking by Local Features

Due to redundancy in images and large variations due to illumination changes and articulation, some pixels can be deemed more informative than others. With this in mind, many tracking applications choose to track specific extracted *keypoints*

from an image, rather than tracking a target by its entire pixel distribution. This has the advantage of offloading many required invariances to the keypoint descriptor, which for many keypoints has been a well studied problem. For example, SIFT keypoints [33] are invariant to both scale and rotation, potentially allowing the author to exclude this complexity from their tracker. Furthermore, tracking by keypoints is relatively robust to partial occlusions when the keypoints are well distributed along the target, as the unoccluded keypoints can still be well matched.

Tracking by keypoints can be separated into two general approaches: methods that find keypoint correspondences and methods that build classifiers with keypoints. Shu et al. [46] build several person-specific SVM classifiers with a feature descriptor containing local binary patterns, colour histograms and HOG features. Kalal et al. [25] train a classifier based on 2-bit binary patterns. In contrast to classifier-based approaches, Zhou et al. [55] present a tracker that fuses information from keypoints and histograms. From a target region of interest in the first frame, a colour histogram and SIFT features are extracted for comparison purposes. Then, starting from the previously tracked location, a local region is searched with mean shift to get a potential tracking result with respect to the colour histogram, and SIFT points are compared to get a potential tracking result from the feature points. Finally, the EM algorithm is used to balance the two results, minimizing the distance between the mean shift tracking result and SIFT correspondences. Wagner et al. [48] use FAST corner detection [42] as features to track and use a forest of spill trees, a variant of k-d trees that are robust against noisy data, to match their descriptors. A motion model is employed to coarsely estimate a search region for previously tracked features for finding correspondences.

One problem with tracking by keypoints is deciding how to update the appearance model to reflect the current appearance of the target. Keypoint trackers generally select keypoints from a template at the beginning of the tracking sequence and progressively track these keypoints, never updating the set from which they match. Hare et al. [21] tackle this problem with structured ouput learning using binary descriptors for efficiency. Each keypoint descriptor is given a dynamically updated weight, which are approximated at the learning stage as binary coefficients of basis vectors used for SVM training. This online learning stage allows the tracker to continuously update the importance of each keypoint through its weight, essentially removing outdated keypoints from the appearance model.

2.4 Sparse Representations

Using a sparse subspace for representing a target's appearance was popularized by Mei et al. [36] and has been frequently adopted in the recent literature [37, 23, 54, 52]. In this approach, a dictionary of reference templates $T = [t_1, ..., t_n] \in \mathbb{R}^{dxn}$ is kept, and new candidate templates are treated as a sparse linear combination of the templates in the dictionary $y \approx Ta$, $a = [a_1, ..., a_n]^T \in \mathcal{R}^n$, with sparsity ensured by ℓ_1 regularization. A candidate's likelihood is evaluated as its reconstruction error, ϵ, the cumulative pixel-wise error between the actual candidate and the transformed linear representation. Errors can be accumulated for a multitude of reasons, such as tracker error, motion blur, and occlusion. To deal specifically with occlusion, *trivial templates* $I = [i_1, ..., i_d] \in \mathbb{R}^{dxd}$ are introduced. A trivial template is a vector with only one non-zero entry, corresponding to an occlusion at that location. Finally, that gives us the following representation of the target as a linear combination of target templates and trivial templates:

$$y = \begin{bmatrix} T & I \end{bmatrix} \begin{bmatrix} a \\ e \end{bmatrix} \tag{2}$$

From that representation, the reconstruction problem is solved by ℓ_1-regularized least squares. Appended to the end of the least-squares equation is a regularization constraint, which encourages the coefficient vectors to have as few non-zero elements as possible while minimizing the prior term for sparse reconstruction. This permits an efficient tracking solution based on a small set of target templates, and can be written as:

$$min|| \begin{bmatrix} T & I \end{bmatrix} \begin{bmatrix} a \\ e \end{bmatrix} ||_2^2 + \lambda|| \begin{bmatrix} a \\ e \end{bmatrix} ||_1 \tag{3}$$

Zhang et al. [52] expand on this method of sparse representation by assumption that there are valuable interdependencies between particles in the particle set of a particle filter, and they can be more sparsely represented by a joint representation over the template dictionary. The problem is posed as a multi-task learning problem, where learning the representation of a particle is considered a single task. This learning task reduces into a similar regularization problem over mixed norms, which can be solved by the Accelerated Proximal Gradient method (APG) [38]. APG solves convex optimization problems with non-smooth terms (the $\ell_{p,q}$ mixed norm is a non-smooth term). It achieves a global solution with quadratic convergence, resulting in an accurate and efficient solution.

A collaboration of generative and discriminative models is proposed by Zhong et al. [54], where the discriminative classifier is trained by stacking positive templates and representing them sparsely. Then, a projection matrix S is computed that projects an original feature space of intensity values into a sparse feature space, where the features are coefficients to reconstruct based on the stacked templates. A particle filter is used to select candidates to project into this feature space. A given candidate x is projected into the sparse feature space by $x' = Sx$ and becomes a representation of the training set and sparse coefficients α. The candidate is computed by minimizing α over $||x' - A'\alpha||_2^2 + \lambda||\alpha||$, where A' is the template set projected into the sparse feature space. The discriminative model operates on a pixel-by-pixel classification basis, making it good for non-rectangular objects so that background pixels are not included in the positive instances of the classifier. The generative classifier is modeled by a sparse histogram representation over image patches to effectively account for the spatial appearance model of the target. An overlapping sliding window is used to generate candidates, in contrast to the particle filter used in the discriminative model. A sparse coefficient vector β is computed over each patch by its reconstruction error against the template dictionary, generated from the centers of a k-means clustering of the dictionary. Finally, each sparse coefficient vector is concatenated into a histogram which is used to compare the similarity between an image patch and the template set. These two models both return classification results and reconstruction errors, which are linearly combined to return the final tracking result.

Methods using sparse representation often differ in their template update model. That is, when and how do you update the dictionary of templates such that the appearance model continues to capture the current appearance of a tracked target through rotations, occlusions, illumination changes, and other large changes in the target's appearance? Tracking with fixed templates is doomed to fail in unpredictable environments such as visual surveillance, due to its inadaptability to new, previously unseen appearances. On the other hand, updating the template too often leads to an accumulation of errors, resulting in tracker drift. Mei et al. [36] have initially proposed adding a weight to each template in the dictionary. A template's weight is increased when recently tracked frames are visually similar to that template, and decreased in the opposite fashion. When the tracking result is not sufficiently similar to any one template, the least important template is removed and the current tracked result is added to the dictionary. Jia et al. [23] propose a probabilistic approach that updates old templates slowly, and new templates rapidly. An incremental learning method [41] based on a PCA updates the basis vectors to adapt to appearance change and learns the visual information that the set of targets have in common, which is used to reduce the dimensionality of target reconstruction via the template dictionary. Zhang et al. [52] associate a weight with each template d_i in the dictionary and iteratively update them at each successive frame, based on whether d_i was used in the reconstruction of the tracked target. Templates that are used more frequently are deemed more important and given higher weights. When the particles from a particle filter cannot be represented with low reconstruction error (up to some predefined threshold), the template set is deemed unsatisfactory and the template with the lowest weight is replaced with the current tracking result. This approach updates often enough to ensure the dictionary is representative, while being robust against occlusions and other noise, as these situations will be quickly filtered out due to their low weight.

Although these approaches of using least-squares over a sparse representation have become popular, some approaches are borrowing these ideas and applying them in a non-sparse manner. Li et al. [31] represent a tracked object with an "appearance subspace", spanned by a set of basis samples encoding the appearance distribution. A linear combination of these basis samples are then used to reconstruct candidate samples, and an optimal solution is computed in closed-form with a metric-weighted least-squares solution. This approach borrows many of the ideas from recent successes in

sparse target representation, but differentiate by not searching for a sparse representation and instead using the closed-form solution. Real-time performance is achieved with weighted reservoir sampling, which maintains and updates the appearance subspace while balancing between sample diversity and adaptability in the learning metric.

2.5 Learned Appearance Model

Rather than hand-picking features or pixels that are powerful for tracking, it is becoming increasingly common for approaches to employ automatic feature selection strategies to learn discriminative tracking features over a video sequence or a dataset of training examples. The most successful of such approaches is the use of Adaboost [44], most famously used in object detection [47]. Adaboost is a boosting algorithm that combines several weak (slightly better than random guessing) classifiers into one strong (highly accurate) classifier. In the context of image processing, this allows for several very simple features to be combined into a complex, powerful appearance model.

Avidan [6] treats the tracking problem as a binary classification problem, classifying on a pixel-basis whether a pixel is the target or background. This is achieved with an updating set of weak classifiers, combined into a strong classifier with Adaboost. An ensemble of weak classifiers is maintained to create a confidence map of the pixels in the current frame. Mean-shift is run on the confidence map to locate the peak (that is, the position of the tracked object). Finally, the ensemble is updated by training a new weak classifier on the current frame and adding it to the ensemble. The weak classifiers are trained with an 11-dimensional feature vector extracted from the image, containing local orientation histograms and pixel colours. The appearance model is kept up to date by holding T weak classifiers, and at each frame, throwing out the oldest one and training a new weak classifier on the data from the current frame. This approach is beneficial because it offloads the large training process into a sequence of small, simple learning tasks that can be performed online. If the target to track is known a priori, a large dataset can be used to train an initial classifier offline before tracking even begins.

Among the most prevalent problems with modern trackers is that of tracker drift. If a tracked sequence is a bit off and the appearance model is updated, the appearance model for the target begins to contain a portion of the background, and slowly tends further away from the actual target due to overfitting, by assuming the entire bounding region contains the target. In MILTrack [7], this problem is tackled by bags of bounding regions, rather than a single box containing the target. Traditionally, discriminative classifiers for tracking use the bounding region of a previously tracked location as a single positive training instance, and select several regions away from that to train the background as negative training instances. With even a very small error at the current tracking step, a template update would include part of the background as a positive instance, decreasing discriminative power and contributing to tracking drift. Instead, Babenko et al. [7] propose to use several bounding regions that are very close to the current tracked target location and bag them into one large positive bag. Moving further away from the current tracked location, negative bounding regions are selected to increase discriminativity against the background. Multiple Instance Learning (MIL) is used to deal with the ambiguity in the training data. Instead of instance/label pairs, we have a *bag* of instances/label pairs. A bag is considered *positive* if one or more of its members is positive (that is, at least one bounding box in the bag is the true target). This solves the problem of inherently ambiguous rectangle labeling, caused by small tracking or detection errors, where part of the target is missing or a large portion of the rectangle is background. Each rectangle is then broken down into a feature vector of Haar-like features, and the most discriminative features are selected based on a MIL variant of Adaboost [16, 19]. This algorithm performs well due to its ability to handle ambiguously labeled training examples, by performing classification over bags rather than single suboptimal instances. This algorithm also permits resistance to short-term occlusions when updating the appearance model, due to the extracted bag of examples around the correct region being added to a large set of weak classifiers, which are trained on the features from previous frames which were unoccluded. However, if the target is often occluded by the same object, and for long periods of time, it is possible that the appearance of the occluder is overlearned and the tracker will drift.

Shu et al. [46] propose a part-based tracking-by-detection approach. They describe a system that employs several part-based person-specific SVM classifiers, which are capable of capturing the specific articulations of bodies in a dynamically

Figure 4. Three different detection configurations are considered. Red rectangles correspond to full body detections, green rectangles correspond to upper-body detections, yellow rectangles correspond to head detections.

changing appearance and background. These person-specific SVMs are trained based on the detections in the tracking sequence, and are used to classify the identity of new detections. They are trained with a feature vector of colour histograms and local binary patterns, with positive examples from the tracked trajectory and negative examples from other trajectories to ensure high discriminative power between individuals. This is beneficial in associating trajectories of the same person who enters and exits the camera's field of view multiple times. Initial detections are done with an offline-trained part-based SVM detector, and a greedy bipartite algorithm is used to associate detections into tracked trajectories. Another advantage of this approach is robustness against partial occlusions. When searching for a new detection at location (x_o, y_o), three possible configurations of a human are considered. These configurations correspond to different sets of human body-parts being unoccluded: the entire body, the upper body parts only, or the head only, as can be seen in Figure 4. This avoids an exhaustive combinatorial search over all possible combinations of unoccluded and occluded parts for each individual. The configuration that maximizes a detection score based on HOG features is selected as the candidate tracking location for position (x_o, y_o). Also, the algorithm handles crowded scenes well by using a correlation-based Kalman filter to track the head of an individual when no detection is achieved, due to occlusion or misdetection.

3. Motion Models

Once the target is represented and an appearance model is extracted from a scene, we must locate a similar appearance in subsequent frames and associate them into one continuous tracked trajectory. An exhaustive search over all possible locations and scales at each frame is both infeasible and unneccesary. There exist a small number of motion frameworks of which one appears in almost every recent tracker in the literature. In this section, we give a brief overview of the most common motion models.

Paper	Bayesian Filter	Local Optimization	Detection
Sevilla-Lara et al. [45]	no	yes	no
Oron et al. [39]	yes	no	no
Babenko et al. [7]	no	yes	no
Adam et al. [2]	no	yes	no
Erdem et al. [17]	yes	no	no
Woo Park et al. [49]	yes	no	no
Shu et al. [46]	yes	no	yes
Avidan [6]	yes	yes	no
Mei et al. [36]	yes	no	no
Mei et al. [37]	yes	no	no
Zhang et al. [52]	yes	no	no
Zhong et al. [54]	yes	no	no
Andriyenko et al. [5]	yes	yes	yes
Jia et al. [23]	yes	no	no
Li et al. [31]	yes	no	no
Kalal et al. [25, 26]	no	yes	yes
Yang et al. [50]	no	no	yes
Hare et al. [21]	no	no	yes
Comaniciu et al. [14]	no	yes	no
Kang et al. [27]	no	no	yes
Wagner et al. [48]	no	yes	no
Zhou et al. [55]	no	yes	yes
Liu et al. [32]	yes	no	no

Table 2. Comparison of motion models used in papers

3.1 Local Optimization

A common approach to locating the target in a new image frame is to treat tracking as a global search problem over a spatially smooth function. That is, represent each possible search location as a point on a continuous function space, where the global minimum represents the true target location. Such a search space is commonly solved with mathematical optimization algorithms, such as gradient descent. These approaches rely on the assumption that the gradient descent of the alignment function (function that returns the similarity between the target to track and the candidate location) will eventually reach a global optimum.

Comaniciu et al. [14] define the the alignment function as the distance between two distributions:

$$d(y) = \sqrt{1 - \rho[\hat{p}(y), \hat{q}]} \tag{4}$$

where $\hat{p}(y) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)\hat{q}_u}$ is the sample estimate of the Bhattacharyya coefficient [3] between p and q. As distributions, the colour histograms of the target and candidate are used. The goal is to find the location y that minimizes the proposed distance from the stored target distribution. This is solved using mean shift [13], which is an optimization technique for finding minima of a density function. A confidence map is created based on the distance function and peaks are detected as local minima. These local minima correspond to likely target correspondences. Such a function is made feasible by gradient descent on the distance function, which from some initial point ỹ, returns the next point ỹ' that has the largest drop in distance.

The approach proposed by Comaniciu et al. [14] suffers from a small basin of attraction, which results in the method being more likely to get stuck in a local optimum and not reach the true target. Sevella-Lara et al. [45] propose the use of Distribution Fields to counter this. Distribution Fields perform smoothing both spatially and across intensity histogram bins to achieve a large basin of attraction, which results in gradient descent being much more likely to reach to global optimum of the distance function.

3.2 Bayesian Filters

The Kalman filter and the particle filter are Bayesian filtering frameworks that recursively estimate the current state of the target (for example, location and scale) at each frame. The main advantage of the Kalman filter is that, when the underlying distribution is known to be Gaussian, the Kalman filter provides the optimal tracking solution. However, when the underlying distribution is not known to be a Gaussian, the Kalman filter does not provide the optimal solution and an alternative approach is needed, which gives rise to non-linear models, such as the extended Kalman filter and the particle filter. The extended Kalman filter considers the current mean and covariance estimates of a target's position and tries to linearize the model with these estimates [35]. For highly non-linear observation and prediction models, this yields very poor performance [24], making this unsuitable for most tracking applications. The particle filter was introduced to object tracking as the CONDENSATION algorithm [22]. The particle filter is used to approximate the posterior $p(s_k|z_{1:k})$ by a weighted set of particles $\{s_k^i, \pi_k^i\}_{i=1}^N$ as:

$$p(s_k|z_{1:k}) \approx \sum_{i=1}^{N} \pi_k^{(i)} \delta_{s_k^i}(s_k) \tag{5}$$

In this formulation, $z_{1:k}$ represents the observations at frames 1 to k, s_k is a particle representing a possible state, and π_k is a weight (probability) corresponding to that state.

The method for recursively estimating the current state is achieved in two steps: prediction and update. Prediction consists of generating new particles from the old particle set by importance sampling, where each particle is selected (with

replacement) with probability proportional to its weight. The update phase consists of computing the current observation z_k and updating the weights of the new particles based on this observation.

$$\pi_k^i \propto \pi_{k-1}^i \frac{p(z_k|s_k^i)p(s_k^i|s_{k-1}^i)}{p(s_k|s_{k-1})} = \pi_{k-1}^i p(z_k|s_k^i) \tag{6}$$

After the prediction and update steps, a tracking decision is computed from the particles by a weighted average.

$$\hat{s}_k = \sum_{i=1}^{N} \pi_k^i s_k^i \tag{7}$$

The particle filter has become the most prevalent framework for modeling motion in tracking, due to its adaptability and computational efficiency. Its popularity can be seen in Table 2. One important note when using particle filters is the tradeoff between the number of particles and computational efficiency. Increasing the number of particles permits more accurate tracking, but requires linearly more computations in the number of particles. In contrast, computational efficiency can be increased by reducing the number of particles, which leads to a smaller region of searched spaces for the target.

Liu et al. [32] demonstrate how real-time tracking can be sustained when using a large number of particles a large number of particles by parallelizing the algorithm in the context of a multi-cue face tracker. The proposed tracker uses three feature sets as cues: colour histograms, edge orientation histograms, and Haar filter histograms. A single reference template is used for comparisons, with the Bhattacharyya metric [3] used to compare the colour histogram and edge histogram cues, and the Euclidean distance used to compare the Haar filter cue. A linear combination of these three cues yields the observational likelihood for a particle, which is used to weight each particle. Two parallelization schemes are proposed in [32]: one that leverages multi-core CPUs and one that makes use of GPUs. These approaches take advantage of the fact that the computation of each particle is independent of the computation of the remaining particles.

The multi-core CPU approach in [32] follows the MapReduce paradigm [15], where the programmer specifies a *map* function that processes a key-value pair to generate a set of intermediate key-value pairs, and a *reduce* function that merges the intermediate values sharing the same key. In the context of a particle filter, the *map* function allocates particles across processor cores for parallel likelihood computation. Each core retrieves a particle, representing a possible candidate target location and scale, and that core computes the set of cues and their similarity to the reference template. Finally, the likelihood for that core is computed. The *reduce* function takes as input all of the likelihoods that each core has computed and computes the final target estimate location, representing a single iteration of the particle filter. The authors claim very efficient tracking with this multi-core parallelized approach. Real-time tracking (less than 50 milliseconds) is achieved with 4000 particles on an 8-core consumer-grade processor, while $10,000$ particles are processed per frame at less than 200 milliseconds.

Liu et al. [32] also propose a parallel particle filter algorithm to be run on the GPU. The GPU's kernel is split up into M blocks, each of which contains N threads, where each individual thread computes the observational likelihood for a single particle. This allows for $M * N$ particles to be computed in parallel. Rather than integrating over all particles to gain a final tracking estimate, the authors have chosen to take a maximum likelihood approach and simply select the particle with the highest observational likelihood. Once each block has completed the computations in its N threads, it selects the particle corresponding to this maximized objective function as its return value. Finally, when all of the blocks have selected their best particle, the particle across all blocks with the largest likelihood is selected as the final tracking solution. The authors have shown that this approach is even more efficient than the multi-core approach. Where the multi-core algorithm was able to process 4000 particles in under 50 milliseconds, the GPU algorithm is able to process 6000 particles in the same time. In another comparison, where the multi-core algorithm processes $10,000$ particles in under 200 milliseconds, the GPU algorithm processes $10,000$ partilces in under 90 milliseconds. These experiments run on a consumer-grade NVIDIA GPU with 12 multiprocessors. Images used for experiments for both algorithms have a resolution of 320 by 240.

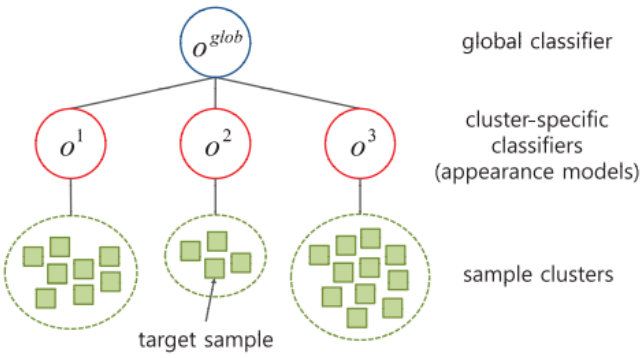

Figure 5. Formerly seen samples are clustered and a cluster-specific classifier is built over each classifier, each contributing to the final tracking decision

Other Bayesian filtering techniques exist besides the Kalman and particle filter. For instance, another sampling technique called Markov Chain Monte Carlo is sometimes used, which samples potential target candidates from a sample distribution, which is iteratively revised in hopes of retrieving the true underlying distribution. This technique is used by Woo Park et al. [49], where rather than simply using time-invariant features like keypoints or histograms, the probabilistic dependency between sequential target appearances is modeled with Autoregressive Hidden Markov Models (AR-HMM). This Bayesian network explicitly defines a probabilistic dependency between the past n sequential target observations, in contrast to standard HMMs which define independence between all observations, and only a dependency between the most recent actual state variable and the current actual state variable. Formerly seen target samples are separated into multiple clusters and a classifier is trained around each cluster, as illustrated in Figure 5. These classifiers independently discriminate the target from the background, and as such represent the appearance model for the target. The Metropolis-Hastings algorithm for Markov Chain Monte Carlo is used as the method to search for a target in the current frame. Potential target locations are drawn from a proposal distribution defined by a Gaussian around the previously tracked location. Although this technique achieves very accurate results, the mean processing time for this algorithm is reported as 8 seconds per frame. The authors note that after several optimizations, real-time tracking is achievable.

3.3 Detection

In the past, running detection algorithms was discouraged due to their high computational cost, so they have been used almost exclusively for seeding trackers with the first frame of the target. As object detectors have become more efficient [47] and computers have become more powerful, the cost of detection is less of an obstacle and many trackers now rely heavily on object detectors to verify tracking results, and are sometimes even used as the entire motion model. When a detector is used as the motion model, the problem of tracking is reduced to finding the optimal labeling of each detection to produce the correct tracked trajectory. We now look at some tracking methods with such a reliance on detectors.

In [50], a multi-target tracking-by-detection approach is modelled with an online-learned conditional random field. Given the detections over a video sequence, a low-level association process [29] connects detection responses in neighbouring frames into reliable *tracklets*. A progressive approach is taken to iteratively connect these tracklets through multiple levels. At each level, the input to the model is the tracklets formed from the previous level, $S = \{T_1, T_2, ..., T_n\}$, and the goal is to find the optimal labelling L where a label $\ell_i \in L$ denotes whether two tracklets should be connected into one continuous sequence. More concretely, the goal is to find the best set of associations (labelling) L^* given a set of

tracklets S.

$$L^* = \arg\max_L P(L|S) = \arg\max_L \frac{1}{Z} \exp\left(-\Psi(L|S)\right) \tag{8}$$

where Z is a normalizing constant and Ψ is the cost function. The cost function is a combination of unary and pairwise energies, which correspond to the linking probabilities between two arbitrary tracklets, and the correlation between pairs of tracklets based on discriminative features learned between them. These energies are minimized to compute the optimal labelling over the CRF. The main advantage of this approach is its ability to accurately distinguish between spatially close targets with similar appearance, due to the learned discriminative features between pairs of detections.

Kalal et al. [25] use a simple base tracker to improve tracking efficiency by learning an object detector based on the simple tracker's trajectory. An object appearance model L_t at time t consists of previously tracked frames from the simple tracker, and an object detector based on efficient 2-bit binary patterns is built on L_t. At each frame, both the object detector and the tracker are run. Detections that are far away from the tracked trajectory are deemed poor and their corresponding region of interest in L_t is pruned away, while the tracked trajectory is added to L_t if it is close to a true detection from the learned object detector. The learned object detector allows for tracker re-initialization when the tracker drifts or leaves and re-enters the scene. The greatest limitation of this approach is the assumption that the target is unique in the scene, and tracker initialization must be done manually by clicking on the target in the first frame.

An extension to this is presented in [26], where the method is optimized for face tracking. An offline-trained face detector is used, rather than the general learned detector, and a "validator" module is introduced to validate the identity of each detected face. Using an offline-learned detector increases efficiency at runtime, as a new detector need not be learned online. With the gained efficiency, the learning then focuses on building the validator. The validator takes as input a face patch, and outputs a confidence that the patch corresponds to a given face. Training the validator is initialized with the first detected frame and bootstrapped with consecutive tracked frames. This approach is robust to partial occlusions and targets leaving and re-entering the camera's field of view.

A discrete-continuous optimization approach [5] considers tracking as two distinct problems. The first problem is assigning the correct identity labels to each detected target with a discrete optimization algorithm. The second problem is to fit a continuous function through the detected locations to approximate the correct target trajectory. Detection labeling is solved by the minimization of an energy term within a pairwise Markov random field, where the vertices correspond to each detection and detections in consecutive time steps that are within some pre-defined threshold distance are connected by an edge. This encourages nearby detections in adjacent frames to belong to the same trajectory. For fitting a trajectory to these labelled detections, cubic B-splines are fit to the detections, whose parameters are estimated by minimizing an energy function that combines two energy terms that make a tradeoff between trajectory smoothness and how well the trajectory fits the labelled detections. Finally, since we have defined a separate energy function for the discrete optimization and the continuous optimization, the two functions are unified into a single energy function for discrete-continuous optimization.

4 TRL Evaluation

To evaluate the technology readiness level of face and person tracking technologies, we use provided experimental results from the current literature. The relevant technology readiness levels in this report are TRL-3 (characteristic proof of concept), TRL-4 (component validation in laboratory) and TRL-5 (component validation in relevant environment), because this technology is reasonably mature but not currently running in a local prototype environment. A description of the different technology readiness levels is available in Table 3. TRL assignments are required for five different environments. Notably, PIL (person in lane), controlled chokepoint, uncontrolled chokepoint, little and dense traffic, and outdoors.

TRL [1]	Description
TRL-9	Actual system 'flight proven' through successful mission operations (over 30)
TRL-8	Actual system completed and 'flight qualified' through test and demonstration
TRL-7	System prototype demonstration in operational environment (pilot)
TRL-6	Component validation in relevant environment (mock-up)
TRL-5	Component validation in relevant environment
TRL-4	Component validation in laboratory environment
TRL-3	Analytical and experimental critical function - characteristic proof of concept
TRL-2	Technology concept / application formulated
TRL-1	Basic principles observed and reported

Table 3. Overview of the different Technology Readiness Levels [1]

4.1 Evaluation methodologies

Not all authors evaluate tracker performance by the same metrics. In this section, we briefly review the necessary details for each metric used in the reported experimental results used for TRL assignment.

4.1.1 PASCAL VOC criterion

One commonly used metric is the PASCAL VOC criterion [18], which evaluates a combination of centering accuracy and scale. It can be written as

$$a_0 = \frac{area(predicted \cap ground\_truth)}{area(predicted \cup ground\_truth)} \tag{9}$$

For each frame, a PASCAL criterion of $a_0 > 0.5$ is deemed to be a successful tracking result. Authors often report the percentage of frames that report a PASCAL criterion of greater than 0.5 as the percentage of frames that were accurately tracked.

4.1.2 Mean ℓ_1 distance

Another common evaluation technique is the mean ℓ_1 distance. This metric, also known as the Manhattan distance, computes the ℓ_1 distance from the center of the tracked bounding box to the center of the ground-truth bounding box. This can be written as

$$\ell_1(p, q) = ||p - q||_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{10}$$

where n corresponds to the number of dimensions, normally 2.

4.1.3 CLEAR MOT

The CLEAR MOT metrics [28] are a standard of evaluation metrics that are sometimes used to compare different trackers. Shu et al. [46] evaluate on TP (tracking precision), TA (tracking accuracy), DP (detection precision) and DA (detection accuracy). Also from this standard, Andriyenko et al. [5] evaluate on MOTA (Multi-Object Tracking Accuracy), which combines missed targets, false alarms, and identity switches into one normalized number between 0 and 100 percent. MOTP (Multi-Object Tracking Precision) evaluates how precise the tracking is, by computing the average overlap between

Paper	Code location
Sevilla-Lara et al. [45]	http://people.cs.umass.edu/~lsevilla/trackingDF.html
Oron et al. [39]	http://www.eng.tau.ac.il/~oron/LOT/LOT.html
Babenko et al. [7]	http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml
Adam et al. [2]	http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm
Erdem et al. [17]	http://web.cs.hacettepe.edu.tr/~erkut/codes/AMFTracker.zip
Shu et al. [46]	Not available
Avidan [6]	Not available
Mei et al. [36]	http://www.dabi.temple.edu/~hbling/code_data.htm
Mei et al. [37]	http://www.dabi.temple.edu/~hbling/code_data.htm
Zhang et al. [52]	Not available
Zhong et al. [54]	http://faculty.ucmerced.edu/mhyang/project/cvpr12_scm.htm
Andriyenko et al. [5]	http://www.gris.tu-darmstadt.de/~aandriye/dctracking.html
Jia et al. [23]	http://faculty.ucmerced.edu/mhyang/project/cvpr12_jia_project.htm
Li et al. [31]	Not available
Kalal et al. [25, 26]	https://github.com/zk00006/OpenTLD
Yang et al. [50]	Not available
Hare et al. [21]	http://www.samhare.net/research/keypoints/code

Table 4. Code availability for the studied approaches

the tracked bounding box and ground truth. Similarly to MOTP, the result is normalized. We now list the remainder of the used CLEAR MOT metrics in this survey.

- *MT* indicates the ratio or number of trajectories for which the target is "mostly tracked", successfully tracked for more than 80% of the sequence.
- ML indicates the ratio or number of trajectories that are "mostly lost", successfully tracked for less than 20% of the sequence.
- FM indicates the number of times that a ground truth trajectory is interrupted.
- IDS indicates the number of times a tracked trajectory switches its target id.

4.2 PIL

The requirements for person tracking in the PIL setup are the most surmountable of the evaluated environments. The camera is at approximately face level and pointing in the direction of the trackable person, making face detection possible for tracker seeding. Furthermore, the scene will not face a great deal of clutter, but must be able to handle rotation of the face and potential self-occlusion by events such as a person raising their hand to their face. Such scenarios are commonly used for experimental results in the literature, and as such many results and comparisons are available. However, it is important to note that all of the following datasets used to evaluate the PIL environment are laboratory setups, meaning the highest assignable TRL is 4.

The first video sequence used for evaluation is the "occluded face" video sequence, provided by Adam et al. [2]. In this sequence, a single face is present and is directly facing the camera. The face is repeatedly occluded with a magazine, at

Approach	Mean ℓ_1 distance (pixels)	% of well tracked frames (PASCAL criterion)
Fragtrack [2]	6.34	NA
MILTrack [7]	27.23	77.28
Distribution Fields [45]	5	100

Table 5. Comparison of performance on the "Occluded face" sequence

Approach	Mean ℓ_1 distance (pixels)	% of well tracked frames (PASCAL criterion)
Fragtrack [2]	45.19	NA
MILTrack [7]	20.19	78.13
Distribution Fields [45]	11.25	98.76

Table 6. Comparison of performance on the "Occluded face 2" sequence

some points occluding over 90% of the face. This video sequence is demonstrative of robustness against partial occlusions, but does not exhibit large appearance changes, such as a turned face, and only one face is present. Fragtrack [2] performs very well on this sequence, showing virtually no tracker drift in light of very heavy occlusions. Babenko et al. [7] performed a comparison showing that Fragtrack had a mean ℓ_1 distances on this sequence of 6.34 pixels, while MILTrack has a mean ℓ_1 distance of 27.23 pixels. Distribution Fields [45] are reported to have a mean ℓ_1 distance of 5 pixels, with 100% of the frames being tracked correctly, in contrast to MILTrack only tracking 77.28% of the frames correctly. These results are organized in Table 5.

Babenko et al. [7] provide a similar video sequence, called "occluded face 2", which is similar to the previous video but with added appearance changes. In this sequence, the target rotates his head, hides his face, puts on and takes off a hat, and occludes his face repeatedly with a textbook. Fragtrack tends to fail when the target rotates its head, while MILTrack learns the new appearance and continues to track accurately. Reported results show that Fragtrack has a mean ℓ_1 distance of 45.19 pixels, while MILTrack has a mean ℓ_1 distance of 20.19 pixels. Despite 20.19 pixels seeming like a large error, examination of the tracking result shows that the tracker does not stray from the actual target. Distribution Fields are reported to have a mean distance of 11.25 and correctly track 98.76%, in contrast to MILTrack correctly tracking 78.13% of the frames. This result is suspect, since observation of MILTrack on this sequence as a human appears to show much more accurate tracking results than 4 of every 5 frames. The results are organized for comparison in Table 6.

Birchfield et al. [11] provide the "girl" video sequence, which has a similar camera setup to the PIL scenario, but the camera pans and zooms with respect to the target. This sequence showcases scale changes, the target rotating and moving around the scene rather quickly, and a secondary human as an occluder. Distribution Fields are reported to correctly track 73% of the frames, while MILTrack reportedly correctly tracks 55.2% of frames. The mean ℓ_1 distance for Distribution Fields is 21.57 pixels, compared to 32.76 pixels for MILTrack and 16 pixels for Fragtrack with adaptive cues. Mei et al. [37] and Zhang et al. [52] have not released exact numbers, but graphically show that their ℓ_1 tracker and multi-task sparse tracker (respectively) achieve a more accurate mean ℓ_1 distance than MILTrack as well, with a mean error of approximately 10 pixels. The sparsity-based collaborative tracker [54] appears to have achieved the state of the art on this sequence, yielding a mean ℓ_1 distance of just 9.8 pixels. This approach is worthy of further investigation, as it reports very significant improvements over the state of the art on almost all standard evaluation sequences. The reported comparable results are organized in Table 7.

Approach	Mean ℓ_1 distance (pixels)	% of well tracked frames (PASCAL criterion)
Fragtrack with Adaptive Cues [17]	16	NA
MILTrack [7]	32.76	55.2
Distribution Fields [45]	21.57	73
Multi-task Sparse Learner [52]	10	NA
ℓ_1 tracker [37]	10	NA
Collaborative Tracker [54]	9.8	NA

Table 7. Comparison of performance on the "Girl" sequence

4.2.1 Evaluation

Several demonstrated trackers perform well on datasets that are relatable to the PIL environment. Specifically, the Distribution Fields tracker [45] achieves very accurate results on all three evaluated video sequences, both in terms of the mean distance to the target center and the number of frames that the target is well-tracked in. We remind the reader that, although these techniques have very favourable performance results, the tested video sequences are devised in a laboratory environment. As such, we propose a TRL of *4* for person tracking in the PIL environment.

4.3 Controlled chokepoint

A controlled chokepoint can be defined as a scene where a doorway entrance is monitored and a small number of individuals are passing through it in controlled intervals. Such a scenario could possibly be the exit of an airport security checkpoint. Due to the controlled point-of-entry, a face or body detection is achievable to seed the tracker. The tracker must be robust to events such as rotations and transformations, such as a person crouching to tie their shoe or secure their luggage. Unfortunately, there are no relevant video sequences that are commonly evaluated against in the current literature. However, if a relevant dataset is available, the source code for many of these trackers is available for evaluation, and the location of each tracker's available source code is noted in Table 4.

4.4 Uncontrolled chokepoint

An uncontrolled chokepoint is very similar to the previously mentioned controlled chokepoint, but there are no requirements on the number of people entering at a time, or the manner by which they enter. Face or body detection might not be immediately achievable due to possible occlusions. Unfortunately, there are no relevant video sequences that are commonly evaluated against in the current literature. However, if a relevant dataset is available, the source code for many of these trackers is available for evaluation, and the location of each tracker's available source code is noted in Table 4.

4.5 Little/Dense Traffic

The requirements for person tracking in the Little/Dense Traffic environment are much more rigid than the previously described setups. Videos must contain several humans entering and exiting the camera's field of view, often occluding one another. A tracker must be able to handle multiple targets, exits and entrances of targets, partial and full occlusion, and significant appearance changes, as humans are able to move freely with all parts of their bodies. We have noted a small number of videos that are evaluated in the recent literature that fit this criteria in a relevant environment. The maximum assignable TRL for these videos is 5.

Approach	% of well tracked frames (PASCAL criterion)
Locally Orderless Tracker [39]	69.6
MILTrack [7]	2.9

Table 8. Comparison of performance on the "PETS-2006" sequence

Approach	Mean ℓ_1 distance (pixels)	% of well tracked frames (PASCAL criterion)
Fragtrack [2]	116.1	19
MILTrack [7]	100.2	19
ℓ_1 tracker [37]	65.7	20
Collaborative Tracker [54]	2.7	NA
Adaptive Structural Local Sparse Tracker [23]	2.3	84

Table 9. Comparison of performance on the "Caviar" sequence

A relevant video sequence for evaluating this style of tracking is a 900 frame video of a train stop from the PETS-2006 dataset. The video shows travellers walking through a stop at a train station. This video exhibits many changes in pose and many occlusions. Oron et al. [39] report that Locally Orderless Tracking is the current state of the art on this dataset, providing good tracking results on 69.6% of the frames by the PASCAL VOC criterion. In contrast, MILTrack only returned successful tracking on 2.9% of the frames. It is suggested that the occlusions present early on in this sequence cause other evaluated trackers to drift and not recover, while Locally Orderless Tracking continues tracking the target.

The *caviar* sequence is video footage from a surveillance camera tracking several humans walking through a hallway with doors to exit into along the side of the camera's field of view. Several humans overlap and occlude one another, stop and interact, and enter and exit the scene from different locations. Zhong et al. [54] report a quantitative analysis of several trackers on this sequence with a tracking metric of mean ℓ_1 distance. Two recent approaches report significant gains on this dataset. The collaborative model sparse tracker presented by Zhong et al. reports significant improvements, with a reported error of 2.7 pixels. The adaptive structural local sparse tracker by Jia et al. [23] also reports very accurate results, with an error of 2.3 pixels. In contrast, the ℓ_1 tracker presented by Mei et al. [36] has an average distance of 65.7 pixels, MILTrack has an average distance of 100.2 pixels, and Fragtrack has an average distance of 116.1 pixels. The disparity between the best trackers and the others is attributed to the generative model by [54], that reduces the impact of occlusions, resulting in a simple foreground histogram comparison. [23] reports similar reasoning, that using both the holistic and local information allows the tracker to not drift when the target disappears and reappears due to short-term full occlusion. Early in the video sequence, a full occlusion of the target causes all other trackers to drift, while the two improved trackers permit immediate tracker recovery.

Shu et al. [46] present a new dataset from airport surveillance footage and evaluate their part-based tracker on this dataset with the CLEAR MOT metric. This sequence has a dense neighbourhood of humans undergoing significant appearance and pose change throughout the video, with a low frame rate and high image resolution (5 fps and 4000 x 2672, respectively). This approach yields a tracking precision of 67.2 and a tracking accuracy of 52.2.

4.5.1 Evaluation

The most characteristic sequence for the Little/Dense Traffic scenario is the caviar dataset. Recent results in [23] and [54] report significant and promising results on this dataset, suggesting that these approaches would be well suited to this environment. Based on these results, we propose a TRL of *5* for person tracking in the Little/Dense environment.

Approach	MT	ML	FM	IDS
Discrete-Continuous MTT [5]	6	0	1	4
CRF MTT [50]	70%	0	1	0

Table 10. Comparison of performance on the "TUD" dataset

4.6 Outdoors

The Outdoors environment presents the most difficult tracking setup of the researched environments. Video sequences may contain any number (including 0) of targets at any given time, with no restrictions on occlusions or illumination changes, target entry and exit at arbitrary times from arbitrary locations, and several available appearance changes due to adding or removal of attire, human articulation, or rotation. We present experimental results on small number of videos that are relevant to this environment. The maximum assignable TRL for these videos is 5.

Andriyenko et al. [5] present experimental results on the *PETS-2009 S2.L1* dataset and the *TUD* dataset. These video sequences show pedestrians walking in an outdoor rural environment. These sequences exhibit several occlusions from overlapping pedestrians due to the low camera angle. The authors compare their results on the PETS-2009 dataset against the previous state of the art [10] as evaluated by the PETS organizers, and show that they have achieved favourable results. By the CLEAR MOT metric, this approach achieves a MOTA score of 89.3 %, comparing favourably to the previous best of 82 % on this dataset. From this result, we read that the approach yields a low number of false alarms, missed targets, or identity switches. However, it says little about how precise the tracking is. Another CLEAR MOT metric has been compared for this purpose. On this same dataset, the authors report a MOTP result of 56.4%, indicating that the tracking could be much more precise, and on average deviates from the true center of the target by almost half of its area. The reported state of the art on this metric, as reported by the PETS organizers, is 60 %.

Yang et al. [50] also report experimental results on the TUD dataset. Due to a small difference in evaluation methodologies used, there is one disparity in comparing [50] against [5]. The "mostly tracked" value for [5] is reported as nominal value, while it is reported as a percentage of the trajectories that are mostly tracked in [50].

4.6.1 Evaluation

It is difficult to evaluate the readiness of these tracking methods to the Outdoors environment, due to the fact that most recent trackers do not evaluate on relevant video sequences for this setup. Of the two trackers that have, [50] appears to have achieved the current state of the art, which accurately tracks 70% of the target trajectories. A more thorough evaluation on more data is recommended before declaring these approaches to have been evaluated in a relevant environment, as there are surely situations that are not captured within these few data sequences that will affect the long-term performance of these trackers. As such, it can be noted that these approaches have been validated on a small dataset, comparable to that of a laboratory environment, and we propose a TRL of *4*.

5 Conclusion

Throughout this survey, several recent advances in tracking have been introduced and evaluation of these approaches to person/face tracking in a visual surveillance environment has been performed in Section 4. From the experimental results provided by the literature introducing these tracking methods, it has been shown that the technology may be mature enough for certain surveillance environments, such as person-in-lane tracking and little/dense traffic. Specifically, our investigations show that sparse methods are significantly outperforming other appearance models for surveillance tracking.

Scenario	Suggested Tracker	TRL
PIL	Distribution Fields [45]	TRL-4
Little/Dense Traffic	Adaptive Strucutral Local Sparse Tracker [23], Collaborative Model Sparse Tracker [54]	TRL-5
Outdoors	CRF Mult-Target Tracker [50]	TRL-4

Table 11. TRL evaluations and suggested trackers for each environment

We have also found that there are an insufficient number of authors currently publishing results against chokepoint and outdoors surveillance environments. We summarize the final TRL evaluations, along with the suggested trackers for each environment, in Table 11, and summarize the available code for the described approaches in Table 4 for any desired further investigations.

References

[1] Strategic readiness level - the esa science technology development route. *European Space Agency, Advanced Studies and Technology Preparation Division*, 24(5), 2012. 17

[2] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 798–805, Washington, DC, USA, 2006. IEEE Computer Society. 2, 4, 5, 12, 18, 19, 21

[3] F. J. Aherne, N. A. Thacker, and P. Rockett. The Bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4):363–368, 1998. 13, 14

[4] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. CVPR 2012 Open Source Award Winner. 7

[5] A. Andriyenko, K. Schindler, and S. Roth. Discrete-continuous optimization for multi-target tracking. In *CVPR*, 2012. 2, 12, 16, 17, 18, 22

[6] S. Avidan. Ensemble tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):261–271, 2007. 2, 10, 12, 18

[7] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *CVPR*, 2009. 2, 10, 12, 18, 19, 20, 21

[8] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006. 7

[9] S. Belongie and J. Malik. Matching with shape contexts. In *Content-based Access of Image and Video Libraries (CBAIVL)*, pages 20–26, Hilton Head, SC, 2000 2000. 3, 4

[10] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *Proceedings of the 12th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (Winter-PETS)*, pages 1–8, 2009. 22

[11] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 232–, Washington, DC, USA, 1998. IEEE Computer Society. 19

[12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer, 2010. 7

[13] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002. 13

[14] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003. 2, 12, 13

[15] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008. 14

[16] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, June 2007. 10

[17] E. Erdem, S. Dubuisson, and I. Bloch. Fragments based tracking with adaptive cue integration. *Comput. Vis. Image Underst.*, 116(7):827–841, July 2012. 2, 4, 5, 12, 18, 20

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 17

[19] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998. 10

[20] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with ssd. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages 790–797, 2004. 3

[21] S. Hare, A. Saffari, and P. H. S. Torr. Efficient online structured output learning for keypoint-based object tracking. In *CVPR*, 2012. 2, 8, 12, 18

[22] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 13

[23] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012. 2, 3, 8, 9, 12, 18, 21, 23

[24] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997. 13

[25] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1417 –1424, 27 2009-oct. 4 2009. 2, 8, 12, 16, 18

[26] Z. Kalal, K. Mikolajczyk, and J. Matas. Face-tld: Tracking-learning-detection applied to faces. In *ICIP*, pages 3789–3792. IEEE, 2010. 2, 12, 16, 18

[27] J. Kang, I. Cohen, and G. Medioni. Object reacquisition using invariant appearance model. In *In Proc. of 17th Int. Conf. on Pattern Recognition (ICPR04*, 2004. 2, 3, 4, 12

[28] R. Kasturi, D. B. Goldgof, P. Soundararajan, V. Manohar, J. S. Garofolo, R. Bowers, M. Boonstra, V. N. Korzhova, and J. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):319–336, 2009. 17

[29] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *CVPR*, pages 685–692. IEEE, 2010. 15

[30] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *ICCV*, pages 2548–2555. IEEE, 2011. 7

[31] X. Li, C. Shen, Q. Shi, A. R. Dick, and A. van den Hengel. Non-sparse linear representations for visual tracking with online reservoir metric learning. In *CVPR*, 2012. 2, 9, 12, 18

[32] K.-Y. Liu, Y.-H. Li, S. Li, L. Tang, and L. Wang. A new parallel particle filter face tracking method based on heterogeneous system. *Journal of Real-Time Image Processing*, pages 1–11, 2011. 2, 12, 14

[33] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 7, 8

[34] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. In *Proceedings of the British Machine Vision Conference*, September 2003. 3

[35] B. A. McElhoe. An assessment of the navigation and course corrections for a manned flyby of mars or venus. *Aerospace and Electronic Systems, IEEE Transactions*, 1966. 13

[36] X. Mei and H. Ling. Robust visual tracking using ℓ_1 minimization. In *ICCV*, pages 1436–1443, 2009. 2, 3, 8, 9, 12, 18, 21

[37] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient ℓ_1 tracker with occlusion detection. In *CVPR*, pages 1257–1264, 2011. 2, 8, 12, 18, 19, 20, 21

[38] Y. Nesterov. Gradient methods for minimizing composite objective function. CORE Discussion Papers 2007076, Universit catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007. 9

[39] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan. Locally orderless tracking. In *CVPR*, 2012. 2, 6, 12, 18, 21

[40] F. M. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR (1)*, pages 829–836. IEEE Computer Society, 2005. 4, 5

[41] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, 77(1-3):125–141, May 2008. 5, 9

[42] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. 7, 8

[43] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000. 6

[44] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. 10

[45] L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *CVPR*, 2012. 2, 6, 12, 13, 18, 19, 20, 23

[46] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *CVPR*, 2012. 2, 8, 10, 12, 17, 18, 21

[47] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)*, pages 511–518. IEEE Computer Society, 2001. 10, 15

[48] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, pages 57–64, Washington, DC, USA, 2009. IEEE Computer Society. 2, 8, 12

[49] D. woo Park, J. Kwon, and K. M. Lee. Robust visual tracking using autoregressive hidden markov model. In *CVPR*, 2012. 2, 12, 15

[50] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. In *CVPR*, 2012. 2, 12, 15, 18, 22, 23

[51] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), Dec. 2006. 2

[52] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012. 2, 3, 8, 9, 12, 18, 19, 20

[53] F. Zhao, Q. Huang, and W. Gao. Image Matching by Normalized Cross-Correlation. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, 2006. 3

[54] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012. 2, 8, 9, 12, 18, 19, 20, 21, 23

[55] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345 – 352, 2009. Special Issue on Video Analysis. 2, 8, 12